LSB算法

2019年9月28日 18:42

近期老师布置了一项作业，让我们探索LSB及其改进算法，查了一点资料，做个笔记记一下。作者能力非常有限，此笔记仅供自己参考。

LSB基础算法

首先说一下LSB算法，Least Significant Bits，将秘密信息嵌入到载体图像像素值的最低有效位，也称最不显著位，改变这一位置对载体图像的品质影响最小。

LSB算法的基本原理：
对空域的LSB做替换，用来替换LSB的序列就是需要加入的水印信息、水印的数字摘要或者由水印生成的伪随机序列。由于水印信息嵌入的位置是LSB，为了满足水印的不可见性，允许嵌入的水印强度不可能太高。然而针对空域的各种处理，如游程编码前的预处理，会对不显著分量进行一定的压缩，所以LSB算法对这些操作很敏感。因此LSB算法最初是用于脆弱性水印的。

LSB算法基本步骤（选自百度百课，在编写matlab程序时并无参考价值）：
1、将原始载体图像的空域像素值由十进制转换成二进制；
2、用二进制秘密信息中的每一比特信息替换与之相对应的载体数据的最低有效位；
3、将得到的含秘密信息的二进制数据转换为十进制像素值，从而获得含秘密信息的图像。

（以上内容选自百度百科，可能不完全正确，但为了介绍完整，因此没有删减）

通过以上内容我们可以看出，LSB的操作对象是载体图像（在matlab里就是像素值矩阵）和秘密信息数据流，因此本次学习我仅仅关注LSB算法本身，没有在如何将秘密信息处理成数据流以及数据流恢复成秘密信息上花太多时间。这里选用的载体图像为lena512*512像素灰度图，秘密信息为256*256像素位图。

首先是秘密信息的嵌入，C为载体图像，M为秘密信息，C_M为载秘图像

```
function [C_M] = LSB_embed(C, M)
C = uint8(C);
M = uint8(M);
[m, n] = size(C);
l = length(M);
C_M = C;
for i = 1:m
```

```
    for j = 1:n
        if (i - 1) * n + j > l
            break;
        end
        C_M(i, j) = C(i, j) - mod(C(i, j), 2) + M((i - 1) * n + j);
    end
end
```

秘密信息的提取，参数含义同上

```
function [M] = LSB_extract(C_M)
C_M = uint8(C_M);
[m, n] = size(C_M);
M = zeros([1, m * n]);
for i = 1:m
    for j = 1:n
        M((i - 1) * n + j) = mod(C_M(i, j), 2);
    end
end
```

下面是数据的处理、嵌入、提取以及画图验证，要注意秘密信息的长度不能超过载体图像的像素数量。

```
clear;clc;
C = imread('lena512.bmp');
M = imread('ctgu256.bmp');
[m, n] = size(M);
M_data = reshape(M, 1, m * n);
C_M = LSB_embed(C, M_data);
M_data_ = LSB_extract(C_M);
M_ = reshape(M_data_(1:m * n), m, n);
figure();
subplot(2, 2, 1);
imshow(C);
title('载体图像');
subplot(2, 2, 2);
imshow(M);
title('秘密信息');
subplot(2, 2, 3);
imshow(C_M);
title('载密图像');
subplot(2, 2, 4);
imshow(M_);
title('提取出的秘密信息');
```

最后结果

载体图像

秘密信息

载密图像

提取出的秘密信息

由于只是改变了载体图像像素值最低位，像素值改变在-1~+1之间，而人眼并不能有效分辨这么微小的改变，因此人眼无法分辨出载密图像与载体图像之间的差别。

LSB算法改进

在网上查了资料，发现大多数算法从算法安全性、抗噪声以及可嵌入数据量三个方面改进，且并不是单独改进某一个方面。

首先是算法安全性，在需要嵌入重要秘密信息时，可选择将秘密信息加密再嵌入到载体图像，或利用混沌序列控制秘密信息嵌入的位置，此时秘密信息可嵌入到除最低有效位的其它位。在研究安全性时，大多为密码学的知识，与数字图像处理关系不大，因此没有深入探索。

抗噪声方面，在载密图像中加入椒盐噪声后提取秘密信息，可以看到提取出的秘密信息中也有椒盐噪声，但仍可以辨别秘密信息。

载体图像

秘密信息

载密图像

提取出的秘密信息

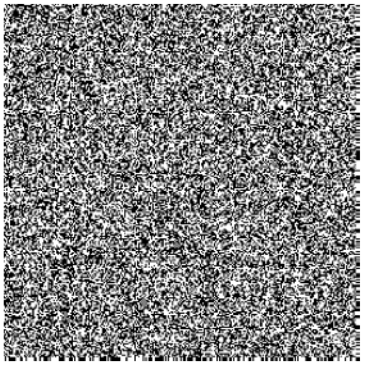
加入均值为0、方差为0.01的高斯噪声后，秘密信息已不能辨认，这里也解释了前面百度百科所说的LSB仅仅适用于脆弱性水印。（加入高斯噪声时，matlab会先将像素值归一化。）

载体图像

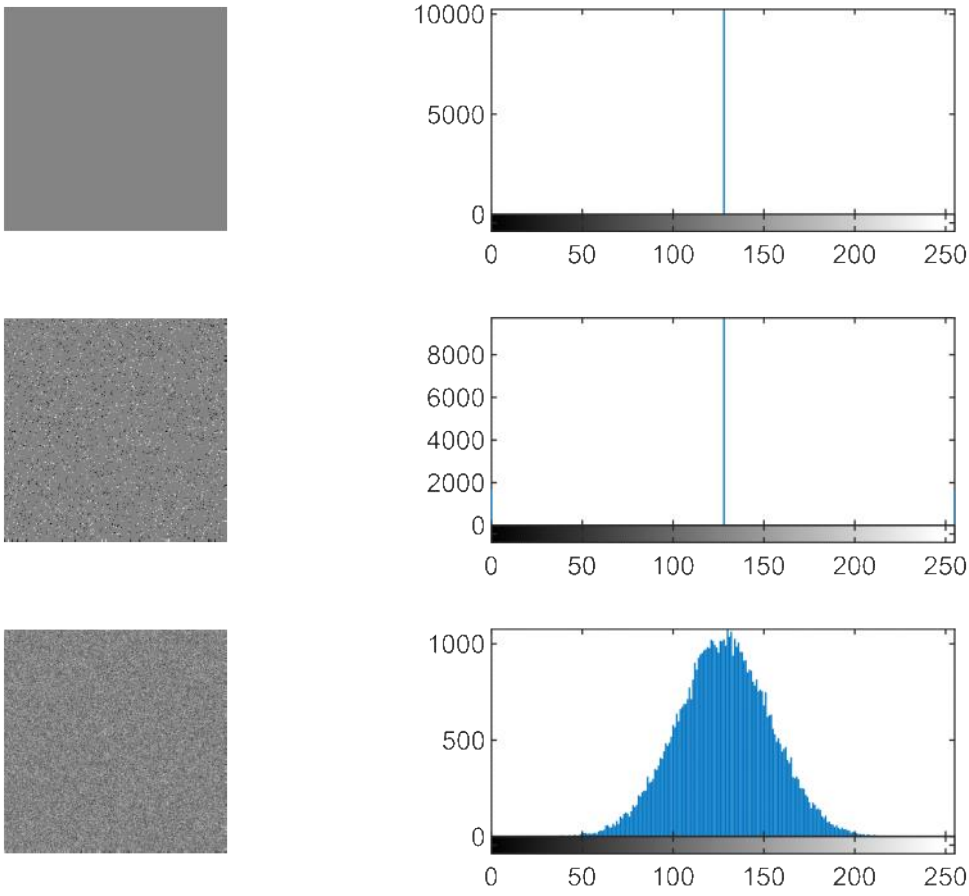
秘密信息

载密图像

提取出的秘密信息



比较一下椒盐噪声与高斯噪声的特性，分别在灰度为128的图片中加入椒盐噪声和高斯噪声后，从直方图可以看出高斯噪声改变的像素点较多，因此对提取出的秘密信息影响更大。



下面讨论如何解决高斯噪声的问题，由高斯分布的特性知道，加入均值为0， 方差为0.01的高斯噪声后，图像像素改变在-25.5~+25.5之间的概率大约为68%，在-49.98~+49.98内的概率大约为96%，因此尝试将秘密信息放在载体图像像素值第七位（规定最低位为第一位）。这里为方便后续研究，将LSB嵌入和提取函数改写，加入描述插入位置的向量作为函数参数。

嵌入函数：

```
function [C_M] = LSB_explore_embed(C, M, B)
C = uint8(C);
M = uint8(M);
[m, n] = size(C);
l = length(M);
C_M = C;
for i = 1:m
    for j = 1:n
        if (i - 1) * n + j > l
            break;
        end
        C_M(i, j) = C(i, j) - mod(C(i, j), 2 ^ B((i - 1) * n + j)) +
M((i - 1) * n + j) * (2 ^ (B((i - 1) * n + j) - 1)) + mod(C(i, j), 2 ^
(B((i - 1) * n + j) - 1));
    end
end
```

提取函数：

```
function [M] = LSB_explore_extract(C_M, B, len)
C_M = uint8(C_M);
```

```
[m, n] = size(C_M);
M = zeros([1, len]);
for i = 1:m
   for j = 1:n
      if (i - 1) * n + j > len
         break;
      end
      M((i - 1) * n + j) = idivide(mod(C_M(i, j), 2 ^ (B((i - 1) * n + j))), 2 ^ (B((i - 1) * n + j) - 1),
'floor');
   end
end
```

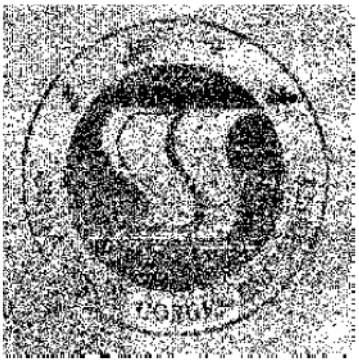
最后结果为

载体图像

秘密信息

载密图像

提取出的秘密信息



可以大致看出秘密信息的轮廓，但是在载秘图片中秘密信息对像素值的影响过大，导致载秘图片变化很大，甚至可以看见秘密信息（可以看见秘密信息是因为没有对秘密信息进行加密，而只是按顺序嵌入）。

通过增大秘密信息对载体图像的权重的方法来抗噪不可取，因此只能考虑用其他的抗噪方法对载秘图像进行处理。

将秘密信息嵌入到载体图像像素值第四位，并使用最简单的平均去噪方法对载密图像进行处理，结果如下

载体图像

秘密信息

100次平均去噪后的载密图像

提取出的秘密信息

可以看到平均去噪对去除高斯噪声的效果非常不错，不仅可以让秘密信息嵌入到载体图像像素值的更低位，也能提高提取出的秘密信息的质量。因此，可以对载秘图像进行去噪处理来提取到更高质量的秘密信息。（平均去噪也有很大的局限性，比如需要传输多幅载秘图像）

最后是可嵌入数据量，其实关于这个的改进并不多，从LSB算法本身来说改进的方面就只有两个，载体图像大小和嵌入方式。可嵌入数据量=载体图像矩阵元素数量*嵌入位数，通过增大载体图像矩阵的元素数量及其像素值的位数，或者增加嵌入的位数（比如嵌入到第一位和第二位），都能增加可嵌入数据量。还可以通过对秘密信息进行压缩等方式间接增大可嵌入数据量。

2019.10.7续

国庆假期没有事情，又看见同学都做了关于LSB的GUI，于是自己也想做一个练练手。考虑到Matlab的GUI对Matlab环境的依赖比较重，在没有Matlab环境时无法使用，带着库打包后也比较大，因此我选择用Python中的Tkinter来做，然后用pyinstaller打包成exe文件，exe文件在dist文件夹中，代码在Github，就不详细介绍了。